LISV Laboratoire d'ingénierie des systèmes de Versailles
UVSQ université PARIS-SACLAY

Architecture des systèmes numériques et informatiques
TD4 : Circuits Logiques Combinatoires

Halim Djerroud

23 octobre 2025

Exercice 1 : Demi-additionneur

On souhaite concevoir un demi-additionneur qui additionne deux bits A et B sans retenue entrante.

1. Établir la table de vérité complète avec les sorties S (somme) et R (retenue).
2. Extraire les expressions booléennes de S et R à partir de la table de vérité.
3. Simplifier l'expression de S en utilisant la porte XOR.
4. Dessiner le logigramme avec :
   — Des portes de base (AND, OR, NOT)
   — Une porte XOR pour la somme
5. Vérifier le fonctionnement pour tous les cas : $(A, B) = (0, 0), (0, 1), (1, 0), (1, 1)$

Exercice 2 : Additionneur complet

Un additionneur complet possède trois entrées : A, B et R_{in} (retenue entrante).

1. Établir la table de vérité complète (8 lignes).
2. Extraire les expressions de S (somme) et R_{out} (retenue sortante) en somme de mintermes.
3. Utiliser des tableaux de Karnaugh pour simplifier :
   — S (vous devriez trouver une fonction XOR triple)
   — R_{out} (fonction majorité)
4. Vérifier que $R_{out} = AB + AR_{in} + BR_{in}$
5. Montrer comment construire un additionneur complet avec deux demi-additionneurs.

Exercice 3 : Additionneur 4 bits

On souhaite additionner deux nombres de 4 bits : $A = A_3A_2A_1A_0$ et $B = B_3B_2B_1B_0$.

1. Dessiner le schéma bloc d'un additionneur 4 bits en cascade (ripple carry).
2. Calculer : $A = 1011_2$ (11) et $B = 0110_2$ (6)
   — Donner l'état de chaque additionneur complet
   — Identifier les retenues intermédiaires
   — Vérifier le résultat final
3. Quel est le problème de l'additionneur en cascade ?
4. Calculer le délai de propagation total si chaque additionneur complet a un délai de 10 ns.
5. Comment pourrait-on accélérer ce circuit ? (Mentionner le concept de retenue anticipée)

Exercice 4 : Comparateur 2 bits

Concevoir un comparateur qui compare deux nombres de 2 bits : $A = A_1A_0$ et $B = B_1B_0$.
Les sorties sont :
— E : égal ($A = B$)
— G : plus grand ($A > B$)
— L : plus petit ($A < B$)

1. Établir la table de vérité complète (16 lignes).
2. Construire les tableaux de Karnaugh pour E, G et L.
3. Simplifier les trois expressions.
4. Vérifier que $E = (A_1 \odot B_1) \cdot (A_0 \odot B_0)$ où $\odot$ est XNOR.
5. Tester votre circuit pour :
   — $A = 11_2$, $B = 10_2$
   — $A = 01_2$, $B = 10_2$
   — $A = 10_2$, $B = 10_2$

Exercice 5 : Multiplexeur 4 :1

Un multiplexeur 4 :1 possède 4 entrées de données (I_0, I_1, I_2, I_3), 2 entrées de sélection (S_1, S_0) et 1 sortie (Y).

1. Établir la table de vérité du multiplexeur.
2. Écrire l'expression booléenne de Y en fonction de S_1, S_0 et des entrées.
3. Dessiner le logigramme complet du multiplexeur.
4. Application : Utiliser un MUX 4 :1 pour implémenter la fonction $F(A, B) = A'B + AB'$ (XOR).
   — Choisir A et B comme entrées de sélection
   — Déterminer les valeurs à connecter sur I_0, I_1, I_2, I_3
5. Généralisation : Peut-on implémenter n'importe quelle fonction de 2 variables avec un MUX 4 :1 ?

Exercice 6 : Démultiplexeur 1 :4

Un démultiplexeur 1 :4 possède 1 entrée de données (I), 2 entrées de sélection (S_1, S_0) et 4 sorties (Y_0, Y_1, Y_2, Y_3).

1. Établir la table de vérité complète.
2. Écrire les expressions booléennes pour chaque sortie.
3. Dessiner le logigramme.
4. Quelle est la relation entre MUX et DEMUX ?
5. Application : Utiliser un DEMUX 1 :4 pour contrôler 4 LEDs avec un seul signal.
   — Comment sélectionner quelle LED allumer ?
   — Peut-on allumer plusieurs LEDs simultanément ?

Exercice 7 : Décodeur 3 :8

Un décodeur 3 :8 convertit un code binaire de 3 bits en 8 sorties (une seule active).

1. Établir la table de vérité pour les entrées E_2, E_1, E_0 et sorties S_0 à S_7.
2. Écrire l'expression de chaque sortie S_i.
3. Comment utiliser un décodeur 3 :8 pour :
   — Adresser 8 emplacements mémoire ?
   — Implémenter n'importe quelle fonction de 3 variables ?
4. Dessiner comment cascader deux décodeurs 3 :8 avec une entrée ENABLE pour créer un décodeur 4 :16.

Exercice 8 : Encodeur prioritaire 4 :2

Un encodeur prioritaire 4 :2 encode 4 entrées en un code binaire de 2 bits, avec gestion de priorités.
Priorités : $E_3 > E_2 > E_1 > E_0$

1. Établir la table de vérité avec :
   — Entrées : E_3, E_2, E_1, E_0
   — Sorties : S_1, S_0 (code binaire) et V (valid)
2. Utiliser des "don't care" (X) pour les entrées de priorité inférieure.
3. Simplifier les expressions avec des K-maps.
4. Que représente la sortie V ?
5. Tester pour :
   — $E_3E_2E_1E_0 = 0000 \rightarrow ?$
   — $E_3E_2E_1E_0 = 0101 \rightarrow ?$ (entrées 2 et 0 actives)
   — $E_3E_2E_1E_0 = 1111 \rightarrow ?$ (toutes actives)

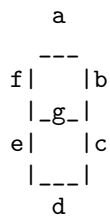
Exercice 9 : Unité Arithmétique et Logique (ALU) simple

Concevoir une ALU 1 bit qui effectue 4 opérations sélectionnées par S_1S_0 :
— 00 : AND ($Y = A \cdot B$)
— 01 : OR ($Y = A + B$)
— 10 : XOR ($Y = A \oplus B$)
— 11 : ADD ($Y = A + B$ avec retenue)

1. Établir la table de vérité complète (4 entrées : A, B, S_1, S_0).
2. Concevoir le circuit en utilisant :
   — Un bloc de portes logiques (AND, OR, XOR)
   — Un additionneur complet
   — Un multiplexeur 4 :1 pour sélectionner la sortie
3. Dessiner le schéma bloc de l'ALU.
4. Comment généraliser cette ALU à n bits ?
5. Quelles autres opérations pourrait-on ajouter ? (NOT, NAND, décalage...)

Exercice 10 : Convertisseur BCD vers 7 segments

Un afficheur 7 segments affiche les chiffres 0-9 en BCD (Binary Coded Decimal).
Les segments sont notés a, b, c, d, e, f, g :



1. Pour les chiffres 0 à 9, établir la table de vérité indiquant quels segments sont allumés.
2. Les codes BCD 10-15 sont invalides. Que devrait afficher le circuit ?
3. Simplifier l'expression du segment a avec un K-map :
   — $a = 1$ pour : 0, 2, 3, 5, 6, 7, 8, 9
   — Utiliser "don't care" (X) pour les codes 10-15
4. Faire de même pour le segment g (barre horizontale centrale).
5. Estimer le nombre total de portes logiques nécessaires pour les 7 segments.

Exercice 11 : Problème de synthèse

On souhaite concevoir un système de contrôle d'ascenseur simple avec 3 étages.
Entrées :
— E_1E_0 : Étage actuel (00, 01, 10 pour étages 0, 1, 2)
— D_1D_0 : Étage de destination
Sorties :
— UP : Monter (1 si destination > position actuelle)
— $DOWN$: Descendre (1 si destination < position actuelle)
— $STOP$: Arrêt (1 si arrivé à destination)

1. Établir la table de vérité complète.
2. Simplifier UP, $DOWN$ et $STOP$ avec des K-maps.
3. Dessiner le circuit logique complet.
4. Vérifier le fonctionnement pour quelques cas :
   — Actuel = 0, Destination = 2
   — Actuel = 2, Destination = 0
   — Actuel = 1, Destination = 1
5. Que se passe-t-il si on code l'étage 2 en binaire avec 11 ? Faut-il gérer ce cas ?

Exercice 12 : Projet - Calculatrice 4 bits

Projet de synthèse : Concevoir une calculatrice simple 4 bits.
Spécifications :
— Deux nombres de 4 bits : $A = A_3A_2A_1A_0$ et $B = B_3B_2B_1B_0$
— Opérations (codées sur 2 bits OP_1OP_0) :
   — 00 : Addition
   — 01 : Soustraction (utiliser complément à 2)
   — 10 : AND bit à bit
   — 11 : OR bit à bit
— Sorties : Résultat 4 bits + Retenue/Borrow + Zero flag
Questions :

1. Dessiner l'architecture globale (schéma bloc).
2. Concevoir l'additionneur/soustracteur 4 bits.
3. Comment détecter un résultat nul (Zero flag) ?
4. Comment implémenter la soustraction avec un additionneur ?
5. Tester avec :
   — $A = 1010$, $B = 0011$, $OP = 00$ (addition)
   — $A = 1100$, $B = 0101$, $OP = 10$ (AND)
6. Estimer le nombre de portes logiques nécessaires.